Noname manuscript No.
(will be inserted by the editor)

A Neuro-interface with Fuzzy Compensator for Controlling Nonholonomic Mobile Robots

Rafiuddin Syam[1], **Keigo Watanabe**[2], **and Kiyotaka Izumi**[2]

[1]Manufacture Division, Department of Mechanical Engineering, Hasanuddin University, Indonesia
e-mail: rafiuddin@internux.web.id
[2]Department of Advanced Systems Control Engineering,
Graduate School of Science and Engineering, Saga University,
1 Honjomachi, Saga 840-8502, Japan
e-mail: watanabe@me.saga-u.ac.jp, izumi@me.saga-u.ac.jp

5 May 2006

Abstract This paper describes a control method for mobile robots represented by a nonlinear dynamical system, which is subjected to an output deviation caused by drastically changed disturbances. We here propose some controllers in the framework of neuro-interface. It is assumed that a neural network (NN)-based feedforward controller is constructed by following the concept of virtual master-slave robot, in which a virtual master robot as a feedforward controller is used to control the slave (i.e., actual) robot. The whole system of the present neuro-interface consists of an NN-based feedforward controller, a feedback PD controller and an adaptive fuzzy feedback compensator. The NN-based feedforward controller is trained offline by using a gradient method, the gains of the PD controller are to be chosen constant, and the adaptive fuzzy compensator is constructed with a simplified fuzzy reasoning. Some simulations are presented to confirm the validity of the present approach, where a nonholonomic mobile robot with two independent driving wheels is assmued to have a disturbance due to the change of mass for the robot.

Key words Neuro-interface, Fuzzy Compensator, Disturbances, Nonholonomic Mobile Robots

1 Introduction

To ease the control of a nonholonomic mobile robot by non-experts, a neurointerface composed by neural network (NN), has been already proposed by Widrow and Lamego [1], [2], which is called WL-neurointerface approach. Such method was already explored by Izumi et al [3] to be very similar to IMC approach [4] and it can also be regarded as one method of the so-called two-degrees-of-freedom (dof) design for robust control. This method is composed mainly of two parts: one is an inverse system realized by an NN to generate a feedforward control input according to a reference value or the output of a reference model and the other is a feedback mechanism to suppress the effect of disturbances due to the change of initial state, mapping errors of NN, etc.

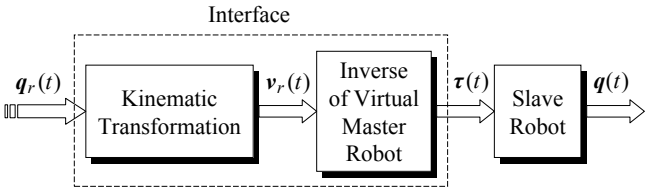

Fig. 1 Construction of an interface using master-slave concept

Note however that Widrow and Lamego[2] provided no systematic ways for constructing NNs for nonholonomic systems, though a usage of any tapped delay inputs was recommended for constructing the NN, and that Izumi et al.[3] also reported a fact that a good result cannot be obtained for a nonholonomic system, even if a feedback error learning mechanism is applied forcedly to acquire an inverse dynamical system. This is attributed to a reason that a unique inverse dynamical system cannot be solved for nonholonomic systems, because most of them include an unstable zero dynamics.

For that reason, Syam et al. [5] have already studied how to derive an NN-based feedforward controler for a nonholomic mobile robot by applying the concept of a vertual master-slave robot, as shown in Fig. 1. It was assumed that there exists an inverse dynamics for a master robot that can be represented by a steering model and that the transformation from the generalized coordinate of a slave robot to the coordinate of a master robot was known. In this research, an NN-based feedforward controller is just as a part of a neurointerface for controlling the nonholonomic mobile robot. Simulation results were also given for the case where the dynamical and kinematic parameters except for the offset distance of steering axis d are all unknown [6].

It is worthy to note that the inverse system acquired by NN will yield a modeling error between it and the ideal model-based inverse system obtained from a known master robot,

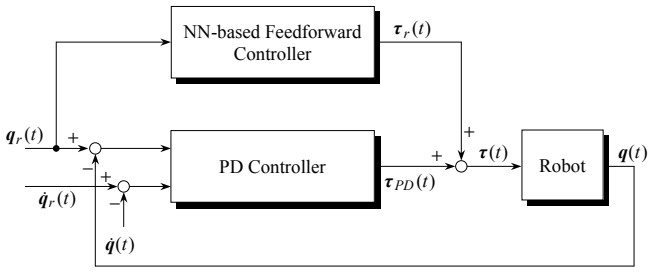

Fig. 2 A neurointerface with a PD feedback controller

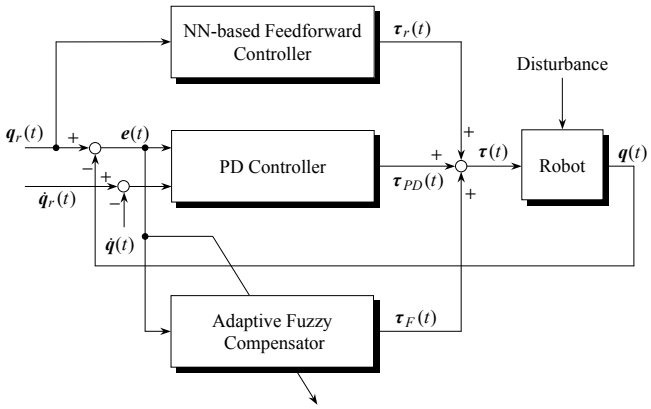

Fig. 3 A neurointerface with a PD feedback controller and an adaptive fuzzy compensator

because a finite amount of output error is normally used to terminate the training process. Therefore, to suppress the effect of such modeling errors, changes of initial configurations (initial disturbances), and unexpected disturbances for test cases, it further needs any feedback compensator.

We have already proposed a neurointerface with a PD feedback controller as shown in Fig. 2 to reduce the effect of mapping error when constructing a feedforward controller through an inverse dynamical model of the virtual master robot. However it should be noted that such a PD controller is effective for a fixed or slowly time-varying environment such as a case where there are no sudden changes of mass of the robot. In this paper, we further consider a case when the mass of the robot will be changed drastically, as a disturbance. In particular, we here introduce an adaptive fuzzy compensator as shown in Fig. 3.

In the following, we first explain a concept of virtual master-slave system in Section 2. In Section 3, we explain the design of some controllers. We first derive an NN-based feedforward controller in the framework of neurointerface, in which a gradient method is used for training the NN obtained through the steering model structure of a master robot. Then, we describe a method with an NN-based feedforward controller and a PD controller as a feedback compensator in the framework of a two-dof design. An adaptive fuzzy feedback compensator is further added to the former method to reduce the effect of drastically changed mass for the robot. In Section 4, several simulation results are presented to illustrate the effectiveness of the proposed methods. Conclusions and further discussions are given in the final section.

2 A Concept of Virtual Master-Slave System

Let the nonlinear dynamical system to be controlled be described by a general nonholonomic two-wheeled robot,

$$\ddot{\boldsymbol{q}}(t) = \boldsymbol{f}_S(\dot{\boldsymbol{q}}(t), \boldsymbol{q}(t), \boldsymbol{\tau}(t)) \tag{1}$$

where $\boldsymbol{q}(t) \triangleq [x(t) \quad y(t) \quad \theta(t)]^T$ is the generalized coordinate vector, in which let the center of mass of the robot be (x, y) and the azimuth of the robot be θ. Moreover, $\boldsymbol{f}_S \in \Re^3$ and $\boldsymbol{\tau} \triangleq [\tau_r(t) \quad \tau_l(t)]^T \in \Re^2$, where τ_r and τ_l are the driving torques of the right and left wheels, respectively. This inverse dynamical model cannot be solved stably and uniquely, so that we further consider the so-called steering dynamical model,

$$\dot{\boldsymbol{v}}(t) = \boldsymbol{f}_M(\boldsymbol{\tau}(t)) \tag{2}$$

where $\boldsymbol{v}(t) \triangleq [v(t) \quad \dot{\theta}(t)]^T$ and $\boldsymbol{f}_M \in \Re^2$, where $v(t)$ denotes the translational velocity of the robot. The inverse of this model is known to be solved stably and uniquely.

Hereafter, it is assumed that the model (1) represents the slave robot to be controlled, while the model (2) represents a master robot to control the slave robot described in Eq. (1).

2.1 Torque Generation by an Inverse Model of Master Robot

In general, we can solve the inverse model of the steering model such as

$$\boldsymbol{\tau}(t) = \boldsymbol{g}_M(\dot{\boldsymbol{v}}(t)) \tag{3}$$

where $\boldsymbol{g}_M \in \Re^2$ is a stable and unique vector-valued inverse function of $\boldsymbol{f}_M$. In order to discretely give $\boldsymbol{v}(t)$ at any time t, we here consider a backward difference model approximation for $\dot{\boldsymbol{v}}(t)$. Then, the above equation can be reduced to

$$\boldsymbol{\tau}(t) = \boldsymbol{g}_M([\boldsymbol{v}(t) - \boldsymbol{v}(t-1)]/\Delta t) \tag{4}$$

where Δt is the sampling width. Given the reference velocity vectors $\boldsymbol{v}_r(t)$ and $\boldsymbol{v}_r(t-1)$ at times t and $t-1$ for the master robot, we can easily obtain the desired input torque vector $\boldsymbol{\tau}_r(t)$ at time t using the above relation.

At this stage, we must be aware of the direct kinematic relation given by

$$\dot{\boldsymbol{q}}(t) = J(\theta(t))\boldsymbol{v}(t),\ J(\theta(t)) = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \tag{5}$$

because the slave robot has its desired reference as $\boldsymbol{q}_r(t)$, where d denotes an offset distance of the center of mass from the mid point of the axle (point P). Therefore, the references $\boldsymbol{v}_r(t)$ and $\boldsymbol{v}_r(t-1)$ for the master robot can be generated by

$$\boldsymbol{v}_r(t) = J^+(\theta_r(t)) \left[\frac{\boldsymbol{q}_r(t) - \boldsymbol{q}_r(t-1)}{\Delta t} \right] \tag{6}$$

$$\boldsymbol{v}_r(t-1) = J^+(\theta_r(t-1)) \left[\frac{\boldsymbol{q}_r(t-1) - \boldsymbol{q}_r(t-2)}{\Delta t} \right] \tag{7}$$

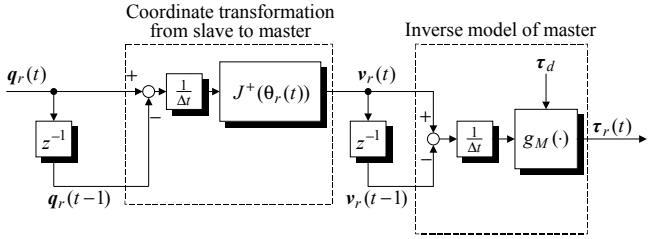

Fig. 4 Model-based feedforward controller with a virtual master-slave system

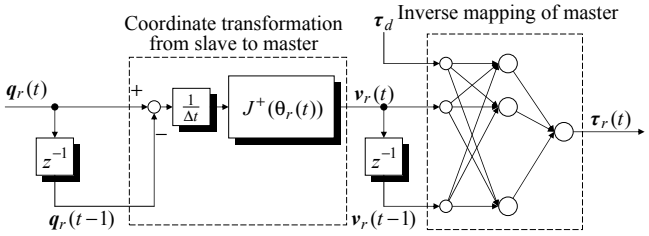

Fig. 5 NN-based feedforward controller with a virtual master-slave system

where $J^+(\cdot)$ denotes the pseudoinverse matrix of $J(\cdot)$.

Thus, given the desired references $\boldsymbol{q}_r(t)$ for the slave robot, we can discretely generate $\boldsymbol{\tau}(t)$ by using Eqs. (4) to (7). Figure 1 shows the construction of an interface using a master-slave concept.

2.2 Addition of a Known Torque Compensation due to Floor Frictions etc.

As can be found from Eq. (4), if $\boldsymbol{v}_r(t)$ is constant, then the desired torque $\boldsymbol{\tau}_r(t)$ will be zero in the sequel. However, in practice, there exist some disturbances such as floor frictions etc., so that we include a small disturbance torque $\boldsymbol{\tau}_d(t)$ to avoid a zero input torque such as

$$\boldsymbol{\tau}_r(t) = \boldsymbol{g}_M([\boldsymbol{v}_r(t) - \boldsymbol{v}_r(t-1)]/\Delta t, \boldsymbol{\tau}_d) \tag{8}$$

Figure 4 shows the block diagram for a feedforward controller by using the inverse dynamical model of a virtual master robot with backward difference approximation, together with a coordinate transformation with a pseudoinverse of a Jacobian matrix. This system is here called a "model-based feedforward controller" to control a slave robot as the final controlled objective.

2.3 Slave and Master Models

We consider the class of nonholonomic mobile robot system having n-dimensional configuration space, with general coordinates $(x_1, \cdots, x_n)$ and subject to m constraints can be described by [7]

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{V}_m(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{\tau}_d \\ = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau} - \boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{\lambda} \end{aligned} \tag{9}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \Re^{n\times n}$ is a symmetric, positive definite inertia matrix, $\boldsymbol{V}_m(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \Re^{n\times n}$ is the centrifugal and Coriolis matrix, $\boldsymbol{F}(\dot{\boldsymbol{q}}) \in \Re^n$ denotes the surface friction, $\boldsymbol{G}(\boldsymbol{q}) \in \Re^n$ is the gravitational vector, $\boldsymbol{\tau}_d \in \Re^n$ denotes bounded unknown disturbances including unstructured and unmodeled dynamics, $\boldsymbol{B}(\boldsymbol{q}) \in \Re^{n\times r}$ is the input transformation matrix, $\boldsymbol{\tau} \in \Re^r$ is the input torque vector, $\boldsymbol{A}(\boldsymbol{q}) \in \Re^{p\times n}$ is the matrix associated with the constraints, and $\boldsymbol{\lambda} \in \Re^p$ is the vector of constraint forces.

For the case discussed in Section 2, we naturally have $n = 3, r = 2$, and $p = 1$, so that it follows that

$$\boldsymbol{M}(\boldsymbol{x}) = \begin{bmatrix} m & 0 & md\sin\theta \\ 0 & m & -md\sin\theta \\ md\sin\theta & -md\sin\theta & I \end{bmatrix},$$

$$\boldsymbol{V}_m(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \begin{bmatrix} 0 & 0 & md\dot{\theta}\cos\theta \\ 0 & 0 & md\dot{\theta}\sin\theta \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{\tau}_d = \boldsymbol{0},$$

$$\boldsymbol{F}(\dot{\boldsymbol{q}}) = \begin{bmatrix} f_x \\ f_y \\ f_\theta \end{bmatrix}, \quad \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{0}, \boldsymbol{\tau} = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix},$$

$$\boldsymbol{B}(\boldsymbol{q}) = \frac{1}{r}\begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ R & -R \end{bmatrix}, \quad \boldsymbol{A}^T(\boldsymbol{q}) = \begin{bmatrix} -\sin\theta \\ \cos\theta \\ -d \end{bmatrix},$$

$$\boldsymbol{\lambda} = -m(\dot{x}\cos\theta + \dot{y}\sin\theta)\dot{\theta}.$$

where m denotes the mass of the robot, I is the moment of inertia for the robot around a vertical axis through the mid point of the axle (point P), $2R$ denotes the tread of the robot, and r denotes the radius of wheel.

The dynamical model (9) as a slave robot, is now transformed into a more appropriate representation for controlling a master robot. That is, the concrete model of Eq. (2) as a master robot can now be reduced as follows:

$$\begin{bmatrix} m & 0 \\ 0 & I - md^2 \end{bmatrix}\begin{bmatrix} \dot{v} \\ \ddot{\theta} \end{bmatrix} + \boldsymbol{\tau}_d = \frac{1}{r}\begin{bmatrix} 1 & 1 \\ R & -R \end{bmatrix}\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{10}$$

3 Architecture Design of Controllers

We here describe several controllers to construct the present neuro-interface. That is, they are composed of an NN-based feedforward controller, PD feedback controllers, and an adaptive fuzzy controller (or compensator), where learning or training algorithms are also presented, if it is necessary.

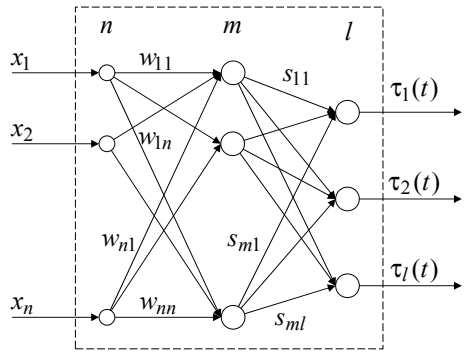

Fig. 6 Three-layered neural network, where all units are assumed to be linear

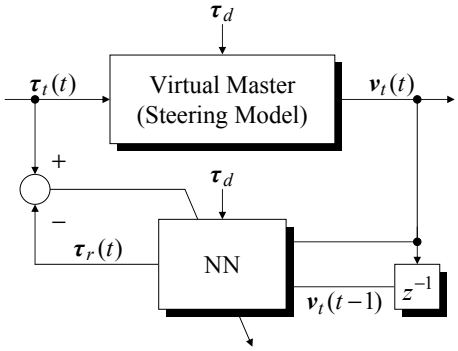

Fig. 7 Training for an NN-based inverse mapping of virtual master robot using generalized learning architecture

3.1 Neural Network-based Feedforward Controller

3.1.1 Structure of Neural Network An NN-based feedforward controller can be easily constructed by replacing the inverse model of master robot with a multi-layered feedforward NN, as shown in Fig. 5.

Now, consider a general multi-layered NN with n inputs, m hidden outputs and l outputs, as shown in Fig. 6, where all units are assumed to be linear, $x_1, \cdots, x_n$ denotes the input signals, $o_1, \cdots, o_m$ are the hidden output signals, and $\tau_1, \cdots, \tau_l$ are the output signals. Note here that as can be seen later, the steering model of the master robot is actually linear with respect to $\boldsymbol{v}(t)$ and $\boldsymbol{\tau}(t)$.

3.1.2 Learning Algorithm The training of the inverse dynamical model for the master robot is assumed to be implemented off-line. In addition, assume that constant disturbance or friction input torques are applied to the training process. Of course, such known torques are used as additional inputs to the inverse mapping due to NN. As shown in Fig. 7, the input torques to the steering model are compared with the NN outputs and the difference errors are to be minimized by adjusting the weights of the NN in the framework of a generalized learning architecture [8].

Here, to train the NN, we consider the minimization of a squared output error such as

$$J = \frac{1}{2}\sum_{k=1}^{l} e_{tk}^2(t), \quad e_{tk}(t) = \tau_{kt}(t) - \tau_k(t) \tag{11}$$

where τ_{kt} denotes the teaching signal for the kth output.

It is further assumed that the weights w_{ij} between the input and hidden layers are unknown, and the weights s_{jk} between the hidden and output layers are known, fixed values. Also, it is easy to find that

$$\tau_k(t) = \sum_{j=1}^{m} s_{jk} o_j, \quad o_j = \sum_{i=1}^{n} w_{ij} x_i$$

Then, the gradient of the cost function J with respect to the weights w_{ij} to be learned is derived as

$$\frac{\partial J}{\partial w_{ij}} = \sum_{k=1}^{l} \frac{\partial J}{\partial e_{tk}} \frac{\partial e_{tk}}{\partial w_{ij}} = \sum_{k=1}^{l} e_{tk} \frac{\partial e_{tk}}{\partial w_{ij}}$$

On the other hand, we can find that

$$\begin{aligned} \frac{\partial e_{tk}}{\partial w_{ij}} &= \frac{\partial e_{tk}}{\partial \tau_k}\frac{\partial \tau_k}{\partial w_{ij}} = -\frac{\partial \tau_k}{\partial w_{ij}} \\ &= -\frac{\partial \tau_k}{\partial o_j}\frac{\partial o_j}{\partial w_{ij}} \\ &= -s_{jk} x_i \end{aligned}$$

Therefore, the incremental value of learning weights can be defined as

$$\begin{aligned} \Delta w_{ij}(t+1) &= -\eta \frac{\partial J}{\partial w_{ij}} \\ &= \eta x_i \sum_{k=1}^{l} s_{jk} e_k \end{aligned}$$

or

$$w_{ij}(t+1) = w_{ij}(t) + \eta x_i \sum_{k=1}^{l} s_{jk} e_k \tag{12}$$

where η denotes the small learning rate. If we want to accelerate the above algorithm, the following modification is also recommendable:

$$w_{ij}(t+1) = w_{ij}(t) + \eta x_i \sum_{k=1}^{l} s_{jk} e_k + \alpha \Delta w_{ij}(t) \tag{13}$$

where α denotes the momentum (or accelerate) factor such as $0 \le \alpha < 1$. Note also that the above learning algorithm for each weighting can be readily derived by conventional backpropagation algorithm [8] with all linear units [9].

3.2 PD Feedback Controllers

We here introduce two types of PD feedback controllers, depending on which type of input in the slave level is constructed for a standard PD control law.

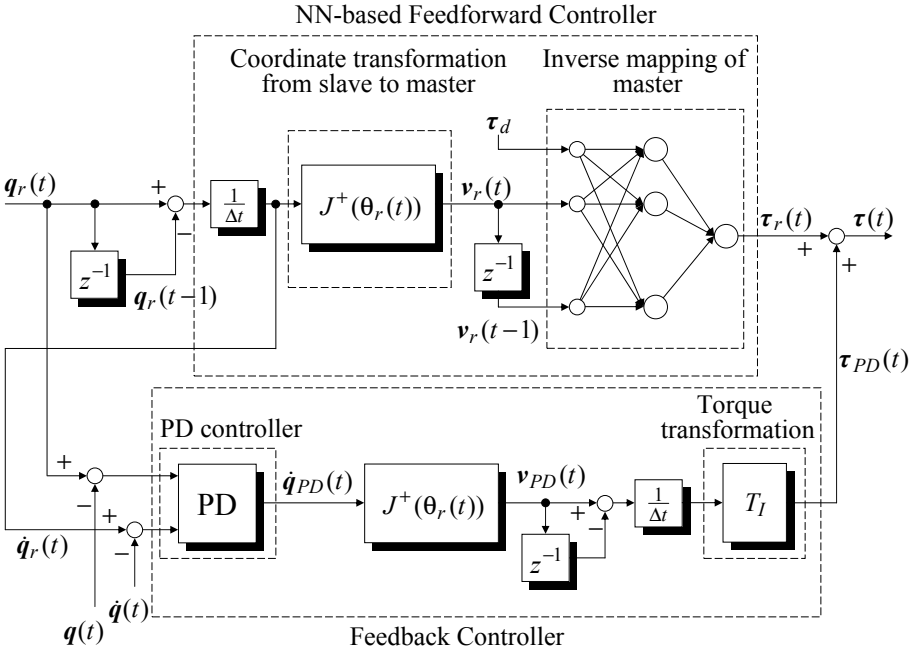

Fig. 8 Neurointerface with a PD feedback compensator

3.2.1 PD feedback controller for type I For the first type, the velocity input in the slave level is assumed to be generated by using a standard PD control law.

If the position error is defined as $\boldsymbol{e}(t) \triangleq \boldsymbol{q}_r(t) - \boldsymbol{q}(t)$, then the velocity input $\dot{\boldsymbol{q}}_{PD}$ is constructed by

$$\dot{\boldsymbol{q}}_{PD}(t) = K_p \boldsymbol{e}(t) + K_d \dot{\boldsymbol{e}}(t) \tag{14}$$

where positive-definite gain matrices are assumed to be $K_p = \text{diag}(k_{1p}, k_{2p}, k_{3p})$ and $K_d = \text{diag}(k_{1d}, k_{2d}, k_{3d})$.

The above velocity input in the slave level can be easily transformed into one in the master level such as

$$\boldsymbol{v}_{PD}(t) = J^+(\theta_r(t))\dot{\boldsymbol{q}}_{PD}(t) \tag{15}$$

Furthermore, the final torque $\boldsymbol{\tau}_{PD}(t)$ can be obtained by the following transformation:

$$\boldsymbol{\tau}_{PD}(t) = T_I \dot{\boldsymbol{v}}_{PD}(t) \tag{16}$$

where it is assumed that $\dot{\boldsymbol{v}}_{PD}(t) = [\boldsymbol{v}_{PD}(t) - \boldsymbol{v}_{PD}(t-1)]/\Delta t$ and T_I is a formal torque transformation matrix whose diagonal element represents any fictitious moment of inertia or mass, e.g., $T_I = \text{diag}(1, 1)$ for the simplicity.

Finally, the total torque $\boldsymbol{\tau}(t)$ from the virtual master robot can be formulated as

$$\boldsymbol{\tau}(t) = \boldsymbol{\tau}_r(t) + \boldsymbol{\tau}_{PD}(t) \tag{17}$$

where $\boldsymbol{\tau}_r(t)$ is a torque generated by the feedforward controller.

Figure 8 shows the block diagram of the proposed neurointerface with a PD feedback controller to suppress the effect of the mapping error etc. in the NN-based feedforward controller.

3.2.2 PD feedback controller for type II For the second type, the acceleration input in the slave level is assumed to be generated by using a standard PD control law. Then, the acceleration input $\ddot{\boldsymbol{q}}_{PD}$ is constructed by

$$\ddot{\boldsymbol{q}}_{PD}(t) = K_p \boldsymbol{e}(t) + K_d \dot{\boldsymbol{e}}(t) \tag{18}$$

This input in the slave level can be transformed into one in the master level such as

$$\dot{\boldsymbol{v}}_{PD}(t) = J^+(\theta_r(t))[\ddot{\boldsymbol{q}}_{PD}(t) - \dot{J}(\theta_r(t))\boldsymbol{v}_{PD}(t)] \tag{19}$$

because of $\dot{\boldsymbol{q}}_{PD}(t) = J(\theta_r(t))\boldsymbol{v}_{PD}(t)$ from Eq. (15) and it is assumed that $\boldsymbol{v}_{PD}(t)$ in Eq. (19) is generated by $\boldsymbol{v}_{PD}(t) = [\Delta t/(1-z^{-1})]\dot{\boldsymbol{v}}_{PD}(t)$. Thus, the final torque $\boldsymbol{\tau}_{PD}(t)$ can be obtained by the same relation as Eq. (16).

Figure 9 shows the block diagram of another PD feedback controller that can be used together with the NN-based feedforward controller.

3.3 Adaptive Fuzzy Compensator

3.3.1 Fuzzy reasoning with a simplified reasoning The simplified reasoning method has an advantage that we can make a fine control, because the conclusion is a function of input data. This method can be interpreted as a special case of the Sugeno's fuzzy reasoning [10]. For n input variables $(e_1, ..., e_n)$ and p output variables $(\tau_{F1}, ..., \tau_{Fp})$ as the consequent, any i-th control rule. In fact, this reasoning method coincides with a case when the function $f_{ij}(e_1, ..., e_n)$ in the conclusion becomes a constant w_{fij}, or a case when the width of the fuzzy set in the conclusion of the min-max-centroidal

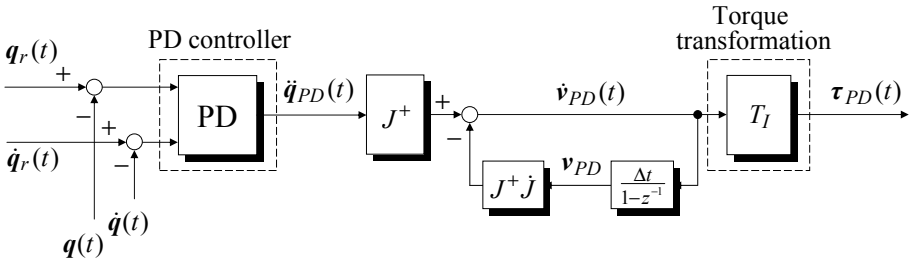

Fig. 9 Another PD feedback controller (type II)

method becomes an infinitesimal value, that is, a singleton. Therefore, any i-th control rule can be written by

$$\begin{aligned} \mathrm{R}_i: \quad & \text{If } e_1 = A_{i1} \text{ and ... and } e_n = A_{in} \\ & \text{then } \tau_{F1} = w_{fi1} \text{ and ... and } \tau_{Fp} = w_{fip} \end{aligned} \tag{20}$$

where R_i denotes the i-th control rule, A_{ij} the fuzzy set (or fuzzy variable) in the antecedent associated with the j-th input variable at the i-th control rule, $f_{ij}(e_1, ..., e_n)$ the function associated with the j-th variable in the conclusion at the i-th control rule. Applying n confidences $\mu_{A_{i1}}(e_1), ..., \mu_{A_{in}}(e_n)$, the confidence in the antecedent h_i is defined by

$$h_i = \mu_{A_{i1}}(e_1) \cdot \mu_{A_{i2}}(e_2) ... \cdot \mu_{A_{in}}(e_n) \tag{21}$$

where w_{fij} denotes a constant associated with the j-th variable in the conclusion at the i-th control rule. Then, following the calculation of the confidence h_i in the antecedent similar to that of (21), the j-th output consequent can be calculated as the following weighted mean of w_{fij} with respect to the weight h_i:

$$\tau^*_{Fj} = \frac{\sum_{i=1}^{r} h_i w_{fij}}{\sum_{i=1}^{r} h_i}, \quad j = 1, ..., p \tag{22}$$

$$\mu_{A_{ij}}(e_j) = \exp\{\ln(0.5)(e_j - c_{ij})^2 w_{dij}^2\} \tag{23}$$

Here, c_{ij} denotes the center value (e.g. the mean value of a Gaussian like membership function) associated with the membership function for the j-th input data at the i-th rule, and w_{dij} denotes the reciprocal value of the deviation from the center c_{ij} to which the Gaussian function of the j-th input data at the i-th rule has value 0.5.

3.3.2 Learning of conseqent part In this section, we derive the learning algorithm of a simplified fuzzy reasoning at the consequent part. To train its part, we consider the minimization of squared output error such as

$$J = \frac{1}{2} \sum_{k=1}^{3} e_k^2(t) \tag{24}$$

where $e_k(t) = q_{rk}(t) - q_k(t)$.

The inceremental value of weights w_{fij} can be expressed as

$$w_{fij}(t+1) = w_{fij}(t) - \eta \frac{\partial J}{\partial w_{fij}(t)} \tag{25}$$

The, the gradient of the cost function J with respect to the weights w_{fij} to be learned is derived as

$$\frac{\partial J}{\partial w_{fij}(t)} = \sum_{k=1}^{3} \frac{\partial J}{\partial e_k(t)} \frac{\partial e_k(t)}{\partial w_{fij}(t)} \tag{26}$$

$$= \sum_{k=1}^{3} e_k(t) \frac{\partial e_k(t)}{\partial q_k(t)} \frac{\partial q_k(t)}{\partial w_{fij}(t)} \tag{27}$$

$$= -\sum_{k=1}^{3} e_k(t) \frac{\partial q_k(t)}{\partial w_{fij}(t)} \tag{28}$$

where,

$$\frac{\partial q_k(t)}{\partial w_{fij}(t)} = \frac{\partial q_k(t)}{\partial \tau_j(t)} \frac{\partial \tau_j(t)}{\partial \tau^*_{Fj}(t)} \frac{\partial \tau^*_{Fj}(t)}{\partial w_{fij}(t)} \tag{29}$$

Since $\tau_j(t) = \tau^*_{Fj}(t) + \tau_{rj}(t)$ and from the fact of (22), we have

$$\frac{\partial q_k(t)}{\partial w_{fij}(t)} = \frac{\partial q_k(t)}{\partial \tau_j(t)} \frac{h_i}{\sum_{i=1}^{r} h_i} \tag{30}$$

Therefore, the incremental value of w_{ij} can be defined as

$$w_{fij}(t+1) = w_{fij}(t) + \eta \sum_{k=1}^{3} e_k(t) \frac{\partial q_k(t)}{\partial \tau_j(t)} \frac{h_i}{\sum_{i=1}^{r} h_i} \tag{31}$$

3.3.3 Learning of antecedent part: case of $c_{ij}(t)$ The incremental value of center parameter c_{ij} can be expressed

$$c_{ij}(t+1) = c_{ij}(t) - \eta \frac{\partial J}{\partial c_{ij}(t)} \tag{32}$$

The gradient of the cost function J with respect to the paramaters c_{ij} to be learned is derived as

$$\frac{\partial J}{\partial c_{ij}(t)} = \sum_{k=1}^{3} \frac{\partial J}{\partial e_k(t)} \frac{\partial e_k(t)}{\partial c_{ij}(t)} \tag{33}$$

$$= \sum_{k=1}^{3} e_k(t) \frac{\partial e_k(t)}{\partial q_k(t)} \frac{\partial q_k(t)}{\partial c_{ij}(t)} \tag{34}$$

$$= -\sum_{k=1}^{3} e_k(t) \frac{\partial q_k(t)}{\partial c_{ij}(t)} \tag{35}$$

where

$$\frac{\partial q_k(t)}{\partial c_{ij}(t)} = \frac{\partial q_k(t)}{\partial \tau_j(t)} \frac{\partial \tau_j(t)}{\partial \tau^*_{Fj}(t)} \frac{\partial \tau^*_{Fj}(t)}{\partial h_i(t)} \frac{\partial h_i(t)}{\partial \mu_{A_{ij}}(e_j)} \frac{\partial \mu_{A_{ij}}(e_j)}{\partial c_{ij}(t)} \tag{36}$$

Since

$$\frac{\partial \tau_j(t)}{\partial \tau^*_{Fj}(t)} = 1 \tag{37}$$

$$\frac{\partial \tau^*_{Fj}(t)}{\partial h_i(t)} = \frac{w_{ij}}{\sum_{i=1}^r h_i} - \frac{\sum_{i=1}^r h_i w_{ij}}{\left(\sum_{i=1}^r h_i\right)^2} \tag{38}$$

$$\frac{\partial h_i(t)}{\partial \mu_{A_{ij}}(e_j)} = \prod_{j=1, i\neq j}^{n} \mu_{ij}(e_j) \tag{39}$$

$$\frac{\partial \mu_{A_{ij}}(e_j)}{\partial c_{ij}(t)} = (-2\ln(0.5)(e_j - c_{ij})w_{dij}^2) \times \exp\{\ln(0.5)(e_j - c_{ij})^2 w_{dij}^2\} \tag{40}$$

we have

$$\frac{\partial q_k(t)}{\partial c_{ij}(t)} = \frac{\partial q_k(t)}{\partial \tau_j(t)} f_c(w_{ij}, e_j, c_{ij}, w_{dij}) \tag{41}$$

in which

$$\begin{aligned} f_c(w_{ij}, e_j, c_{ij}, w_{dij}) = & \left[(-2\ln(0.5)(e_j - c_{ij})w_{dij}^2) \right. \\ & \left. \times \exp\{\ln(0.5)(e_j - c_{ij})^2 w_{dij}^2\}\right] \\ & \times \left[\frac{w_{ij}}{\sum_{i=1}^r h_i} - \frac{\sum_{i=1}^r h_i w_{ij}}{\left(\sum_{i=1}^r h_i\right)^2}\right] \\ & \times \prod_{j=1, i\neq j}^{n} \mu_{ij}(e_j) \end{aligned} \tag{42}$$

Finally, the incremental value of center parameter, c_{ij} can be defined as

$$c_{ij}(t+1) = c_{ij}(t) + \eta \sum_{k=1}^{3} e_k(t) \frac{\partial q_k(t)}{\partial \tau_j(t)} f_c(w_{ij}, e_j, c_{ij}, w_{dij}) \tag{43}$$

3.3.4 Learning of antecedent part: case of $w_{dij}(t)$ Here, we derive learning algorithm of reciprocal value w_{dij} of the deviation Gaussian function. The incremental value of parameter w_{dij} can be expressed as

$$w_{dij}(t+1) = w_{dij}(t) - \eta \frac{\partial J}{\partial w_{dij}(t)} \tag{44}$$

Here, it follows that

$$\frac{\partial J}{\partial w_{dij}(t)} = \sum_{k=1}^{3} \frac{\partial J}{\partial e_k(t)} \frac{\partial e_k(t)}{\partial w_{dij}(t)} \tag{45}$$

$$= \sum_{k=1}^{3} e_k(t) \frac{\partial e_k(t)}{\partial q_k(t)} \frac{\partial q_k(t)}{\partial w_{dij}(t)} \tag{46}$$

$$= -\sum_{k=1}^{3} e_k(t) \frac{\partial q_k(t)}{\partial w_{dij}(t)} \tag{47}$$

where

$$\frac{\partial q_k(t)}{\partial w_{dij}(t)} = \frac{\partial q_k(t)}{\partial \tau_j(t)} \frac{\partial \tau_j(t)}{\partial \tau^*_{Fj}(t)} \frac{\partial \tau^*_{Fj}(t)}{\partial h_i(t)} \frac{\partial h_i(t)}{\partial \mu_{A_{ij}}(e_j)} \frac{\partial \mu_{A_{ij}}(e_j)}{\partial w_{dij}(t)} \tag{48}$$

Since

$$\frac{\partial \mu_{A_{ij}}(e_j)}{\partial w_{dij}(t)} = (2\ln(0.5)(e_j - c_{ij})^2 w_{dij}) \times \exp\{\ln(0.5)(e_j - c_{ij})^2 w_{dij}^2\} \tag{49}$$

we have

$$\frac{\partial q_k(t)}{\partial w_{dij}(t)} = \frac{\partial q_k(t)}{\partial \tau_j(t)} f_{w_d}(w_{ij}, e_j, c_{ij}, w_{dij}) \tag{50}$$

in which

$$\begin{aligned} f_{w_d}(w_{ij}, e_j, c_{ij}, w_{dij}) = & \left[(2\ln(0.5)(e_j - c_{ij})^2 w_{dij}) \right. \\ & \left. \times \exp\{\ln(0.5)(e_j - c_{ij})^2 w_{dij}^2\}\right] \\ & \times \left[\frac{w_{ij}}{\sum_{i=1}^r h_i} - \frac{\sum_{i=1}^r h_i w_{ij}}{\left(\sum_{i=1}^r h_i\right)^2}\right] \\ & \times \prod_{j=1, i\neq j}^{n} \mu_{ij}(e_j) \end{aligned} \tag{51}$$

Finally, the incremental value of reciprocal paramater w_{dij} can be defined as

$$\begin{aligned} & w_{dij}(t+1) \\ & = w_{dij}(t) + \eta \sum_{k=1}^{3} e_k(t) \frac{\partial q_k(t)}{\partial \tau_j(t)} f_{w_d}(w_{ij}, e_j, c_{ij}, w_{dij}) \end{aligned} \tag{52}$$

3.4 An Approximate Evaluation of Output Jacobian with Respect to Input Torque

For the simplicity of the problem, let us consider the case of $d \simeq 0$. Then, it is easily found, from the dynamical model of the slave robot, that

$$M^{-1} = \begin{bmatrix} \frac{1}{m} & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & \frac{1}{I} \end{bmatrix}, \quad M^{-1}B(\boldsymbol{q}) = \begin{bmatrix} \frac{1}{mr}\cos\theta & \frac{1}{mr}\cos\theta \\ \frac{1}{mr}\sin\theta & \frac{1}{mr}\sin\theta \\ \frac{1}{Ir}R & -\frac{1}{Ir}R \end{bmatrix} \tag{53}$$

Therefore, the evaluation of $\partial\ddot{\boldsymbol{q}}/\partial\tau_i(t)$ is equivalent to

$$\frac{\partial M^{-1}B(\boldsymbol{q})\boldsymbol{\tau}(t)}{\partial \tau_1(t)} = \begin{bmatrix} \frac{\cos\theta}{mr} \\ \frac{\sin\theta}{mr} \\ \frac{R}{Ir} \end{bmatrix}, \quad \frac{\partial M^{-1}B(\boldsymbol{q})\boldsymbol{\tau}(t)}{\partial \tau_2(t)} = \begin{bmatrix} \frac{\cos\theta}{mr} \\ \frac{\sin\theta}{mr} \\ -\frac{R}{Ir} \end{bmatrix} \tag{54}$$

Now, using an approximation of $\dot{\boldsymbol{q}} = [\boldsymbol{q}(t) - \boldsymbol{q}(t-1)]/\Delta t$ and noting that the evalution of $\partial\boldsymbol{q}(t)/\partial\tau_j(t)$ is equivalent to $\partial(\Delta t)^2 M^{-1}B(\boldsymbol{q})\boldsymbol{\tau}(t)/\partial\tau_j(t)$, we have

Table 1 Physical parameters for the robot

m [kg]	I [kgm^2]	R [m]	r [m]	d [m]
10	5	0.5	0.05	0.2

Table 2 Mass change of each case simulation

Number of case	mass m [kg]
Case 1	10
Case 2	11
Case 3	12
Case 4	13
Case 5	14
Case 6	15

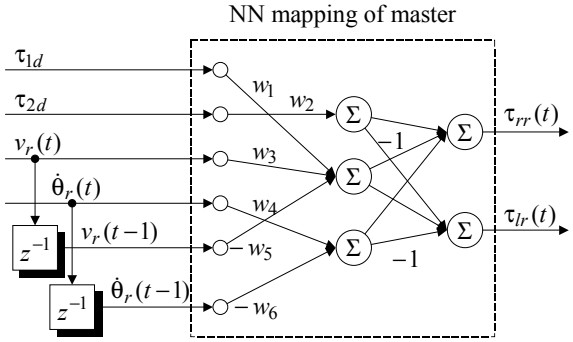

Fig. 10 NN-based inverse mapping of virtual master robot with its dynamical structure, where all the physical parameters are assumed to be unknown except for d

$$\frac{\partial \boldsymbol{q}(t)}{\partial \tau_1(t)} = \begin{bmatrix} (\Delta t)^2 \cos\theta \\ (\Delta t)^2 \sin\theta \\ (\Delta t)^2 R_{mI} R \end{bmatrix}, \quad \frac{\partial \boldsymbol{q}(t)}{\partial \tau_2(t)} = \begin{bmatrix} (\Delta t)^2 \cos\theta \\ (\Delta t)^2 \sin\theta \\ -(\Delta t)^2 R_{mI} R \end{bmatrix} \tag{55}$$

where a new learning rate should be interpreted as $\eta/mr \triangleq \eta'$, and the ratio of mass to the momemt of inertia $R_{mI} \triangleq m/I$ and the tread $2R$ are assumed to be known to use the above evaluation of the output Jacobian with respect to the input torque in the online learning.

4 Simulations

We simulate the neurointerface scheme presented in the previous section to control a nonholonomic mobile robot and compare their performances in solving a trajectory tracking problem. For this purpose, the following three control approaches have been implemented: 1) Method I: an NN-based feedforward controller, 2) Method II: an NN-based feedforward controller and a PD controller, and 3) Method III: an NN-based feedforward controller, a PD controller and an adaptive fuzzy compensator. The vehicle physical parameters used for the robot are tabulated in Table 1 and the sampling width of simulation is $\Delta t = 0.02$ [s]. During the simulation, small disturbance torques that avoid zero input torques were supplied to the NN-based feedforward controller, such as $\{\tau_{1d}, \tau_{2d}\} = \{0.005, 0.005\}$ [Nm] for all simulations.

We simulated six cases of simulations as shown in Table 2, where the mass of the mobile robot was changed for each case of simulations. For example, case 1 denotes the nominal mass of the robot, case 2 is for the mass changed up to 10% from the nominal mass, and case 6 is for the mass changed up to 50% from the nominal mass.

4.1 Method I

We conducted the training of NN to obtain an inverse mapping of the virtual master robot and after that implemented the neurointerface as shown in Fig. 1 for controlling the actual (slave) robot. The steering model of Eq. (10) and the slave model given in Eq. (9) were all simulated by using a simple Euler's method. Note that the initial values of connection weights w_i were set by using uniform random numbers.

We collected the input-output data from the steering model with deterministic input torques, as the training data for the NN. That is, it was assumed that the torque inputs were generated by using the following trigonometric functions:

$$\tau_r = \sin(t)$$
$$\tau_l = \cos(t)$$

for $0 < t \leq 10$ [s].

In this simulation, we considered a case where the dynamical and kinematic parameters, except for the offset distance d of the center of mass from the mid point of the axle (point P), were all unknown.

From Eqs. (3) and (10), we have the following model-based interface given by

$$\begin{aligned} &\boldsymbol{\tau}(t) \\ &= r \begin{bmatrix} 1 & 1 \\ R & -R \end{bmatrix}^{-1} \left\{ \begin{bmatrix} m & 0 \\ 0 & I - md^2 \end{bmatrix} \dot{\boldsymbol{v}}(t) + \boldsymbol{\tau}_d \right\} \\ &= \begin{bmatrix} \frac{1}{2} mr\dot{v}(t) + \frac{1}{2} r\tau_{1d} + \frac{r}{2R}(I - md^2)\ddot{\theta}(t) + \frac{r}{2R}\tau_{2d} \\ \frac{1}{2} mr\dot{v}(t) + \frac{1}{2} r\tau_{1d} - \frac{r}{2R}(I - md^2)\ddot{\theta}(t) - \frac{r}{2R}\tau_{2d} \end{bmatrix} \end{aligned} \tag{56}$$

From the above assumption on this case, we can obtain the desired feedforward torque vector $\boldsymbol{\tau}_r = [\tau_{rr} \quad \tau_{lr}]^T$ through the NN structure depicted in Fig. 10, if the steering reference vector $\boldsymbol{v}_r(t) = [v_r(t) \quad \dot{\theta}_r(t)]^T$ and their one-delayed vector are given.

Note here that $w_i, i = 1, \ldots, 6$ are the connection weights between the input and hidden units, and to be learned, where their ideal values are as follows: $w_1 = 0.5r, w_2 = 0.5r/R$, $w_3 = 0.5mr/\Delta t, w_4 = 0.5r(I - md^2)/(R\Delta t)$, $w_5 = w_3$, and $w_6 = w_4$. Here, weights between the first and third hidden units and the second output unit are should be -1; others are all to be 1.

This NN with all linear units can be trained to obtain an inverse mapping of virtual master robot using generalized

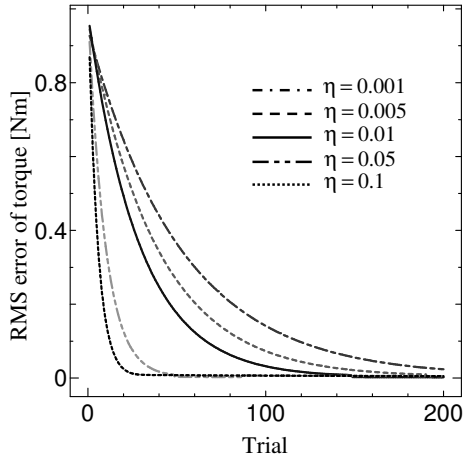

Fig. 11 History of RMS torque error with $\alpha = 0.1$ (Case 1)

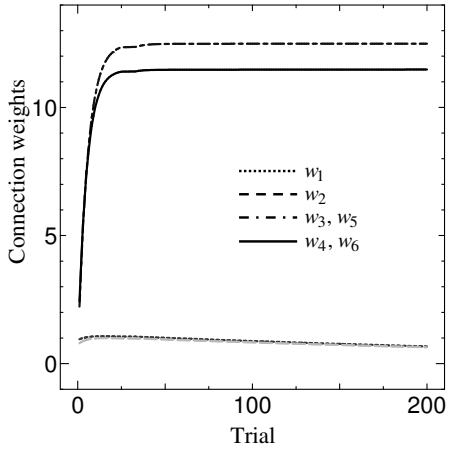

Fig. 12 Learning history of connection weights for $\eta = 0.1$ and $\alpha = 0.1$

Table 3 Learned connection weights

	w_1	w_2	w_3, w_5	w_4, w_6
Ideal	0.025	0.05	12.5	11.5
$\alpha = 0.1$	0.920	1.062	12.487	11.468

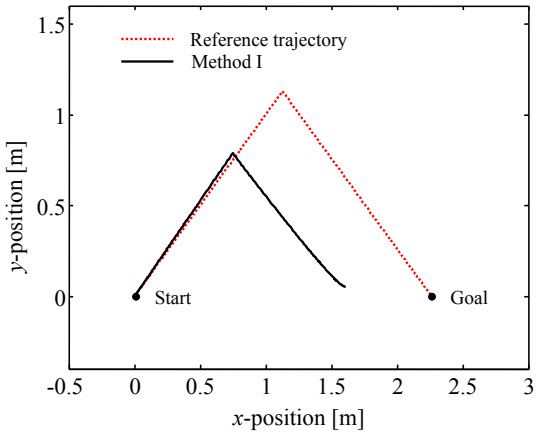

Fig. 13 Trajectory for the mass change of 50% from the nominal mass of the robot (Method I)

learning architecture, as shown in Fig. 7, in which the training data set of $\{\boldsymbol{\tau}_t(t), \boldsymbol{v}_t(t)\}$ are assumed to be collected in advance from an expert operation of the master robot.

We selected several values of learning rate, $0.001 \leq \eta \leq 0.1$ and fixed $\alpha = 0.1$. The training algorithm converged from 30 trials to 200 trials, depending on the selected learning rate as shown in Figs. 11 and 12. Of course, the larger learning rate η can reduce the number of trials, but there are no guarantees to solve the stable convergence problem. On the contrary, a very small learning rate will take expensive computationally. The learned weights with $\eta = 0.1$ can be tabulated in Table 3.

Figures 13 shows a test trajectory tracking of nonholonomic mobile robot, under the condition that the NN-based feedforward controller was trained for the master robot with nominal mass, $m = 10$, but the actual (slave) robot has received an external disturbance caused by drastically making the mass change from the nominal mass to a 50% increased mass. It is seen from this figure that the actual robot was not able to follow the reference trajectory; there was a significant error of the robot position deviated from the reference trajectory.

The time histories of x- and y- positions and azimuth errors are shown in Fig. 14, 15, and 16 respectively, and the resultant RMS error can be found in Figs. 17–19. Since this Method I is concerned only with obtaining a feedforward control input through an inverse system, this controller can not produce an adequate input torque to suppress the effect of any disturbance just like a change of mass.

4.2 Method II

For the second method, we tried to add a PD controller for reducing the disturbance error of the mobile robot. The control system used in this method is shown in Fig. 2, where the PD feedback controller in type I as depicted in Fig. 8 was implemented actually. Here, we have chosen $k_{ip} = 10$ and $k_{id} = 25$ for $i = 1, 2, 3$, as the proportional and differential gains respectively. For the same situation as considered in Method I, the trajectory tracking of the robot can be found in Fig. 20. Observe that this method can not fully reduce the position and azimuth errors of the robot, but its control performance is better than the first method (see also Figs. 14, 15, 16). This fact can also be confirmed from Figs. 17–19.

4.3 Method III

This method can be interpreted as an extended version of Method II. That is, we added an adaptive fuzzy compensator for constructing this control system as shown in Fig. 3.

We here used the simplified fuzzy reasoning where the consequent part was constant value w_{fij}, to construct the adaptive fuzzy compensator. Three input variables were x-position error e_x, y-position error e_y and azimuth error e_θ, and two output variables were the compensated torques τ_{fr} and τ_{fl} for the right and left torques, respectively. Then, the

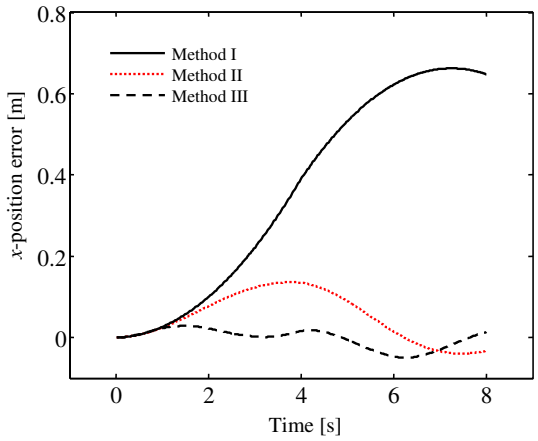

Fig. 14 Error history of x-position for the mass change of 50% from the nominal mass of the robot

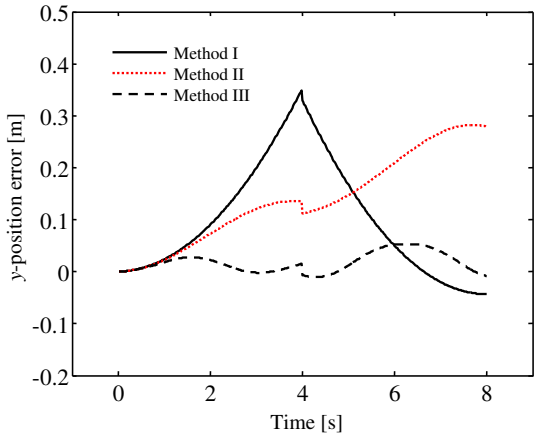

Fig. 15 Error history of y-position for the mass change of 50% from the nominal mass of the robot

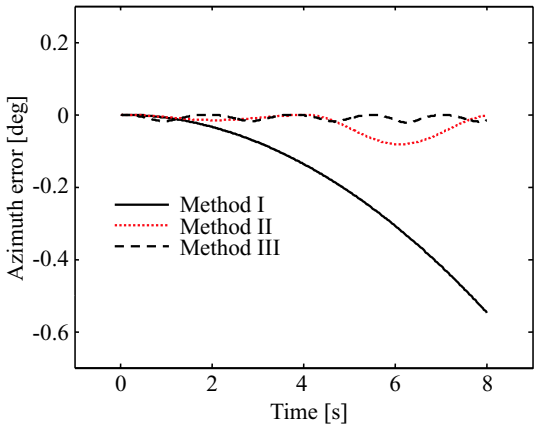

Fig. 16 Error history of θ-azimuth for the mass change of 50% from the nominal mass of the robot

total number of rules was used set to 54 rules, if each input variable has three membership functions. The learning parameter in the fuzzy compensator was only in the consequent part, w_{fij}. That is, we only learned the consequent part, w_{fij} in an online manner. Therefore, all parameters in the antecedent part, i.e., the center value c_{ij} and the reciprocal value

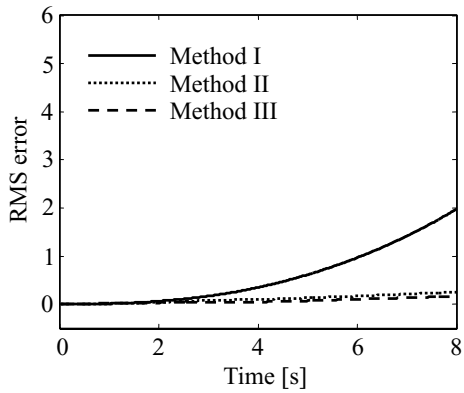

Fig. 17 RMS error history for the nominal mass value of the slave robot

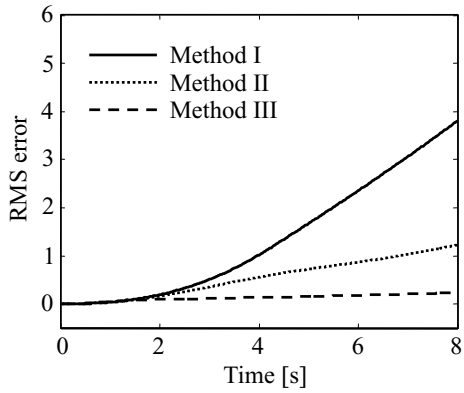

Fig. 18 RMS error history for the mass change of 30% from the nominal mass value

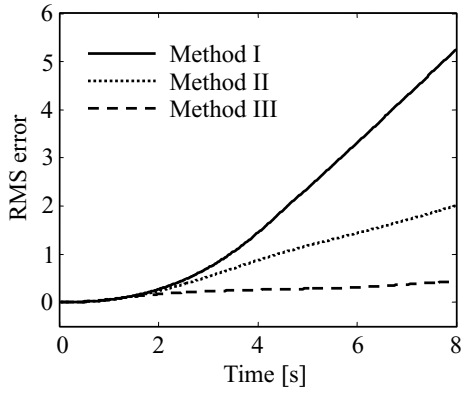

Fig. 19 RMS error history for the mass change of 50% from the nominal mass value

of deviation w_{dij}, were assumed to be unchanged during the online compensation.

The trajectory tracking of the robot using the third method can be found in the Fig. 21. See Figs. 14, 15, 16 for the resultant errors of the robot positions and orientation, and Figs. 17–19 for RMS errors. From these figures, it is seen that as expected, the third method is superior to other two methods.

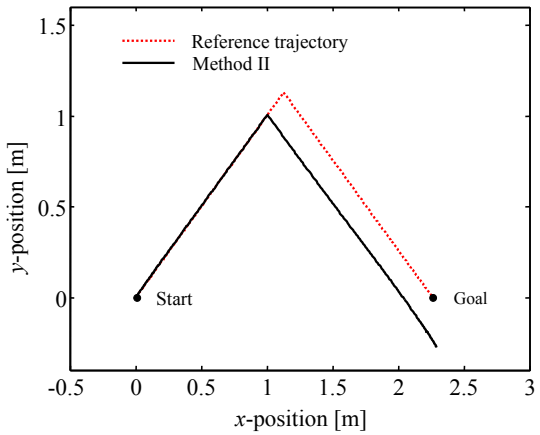

Fig. 20 Trajectory for the mass change of 50% from the nominal mass of the robot (Method II)

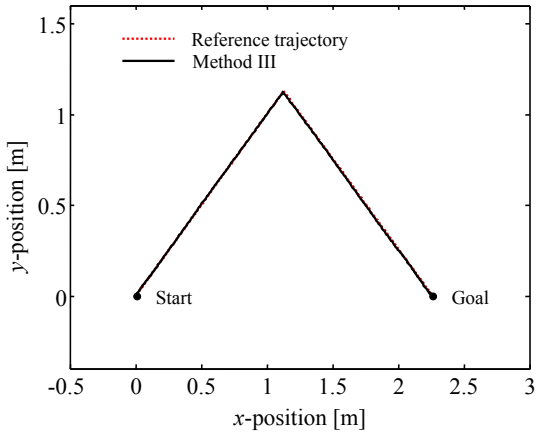

Fig. 21 Trajectory for the mass change of 50% from the nominal mass of the robot (Method III)

5 Discussion

Each case of simulations was conducted by using three methods of neurointerfaces presented in Section 3. The RMS errors were tabulated in Table 4.

The first method is an NN-based feedforward control. This method tried to generate the control input according to the concept of inverse control system, as if there are no any disturbances in controlling the actual (slave) robot. If some additional disturbances were included in the actual robot, then it was found that the controller was not able to cover the position and azimuth errors of the robot in the trajectory tracking problem. The controller was mapped as an inverse of the dynamical model of a virtual master robot with unknown parameters, but they were assumed to be nominal. Note also that the NN-based controller was trained offline. Therefore, this method should be used under an ideal environment without any disturbance, or together with any feedback controller in the framework of two-dof design for robust control.

In the second method, the NN-based feedforward controller that has been obtained in the first method and a PD feedback controller were combined each other. As is well known, the PD controller has a shortcoming in selecting gain

Table 4 RMS error for simulations

	RMS Error		
	Method I	Method II	Method III
Case 1 (nominal)	1.9754	0.2538	0.1618
Case 2	2.2373	0.4185	0.1656
Case 3	2.9994	0.8140	0.2058
Case 4	3.8153	1.2232	0.2319
Case 5	4.5737	1.6228	0.3319
Case 6	5.2561	2.0072	0.4306

factors. We retained the same gains for all simulation cases to explore the performance range of the PD controller with fixed gains. It is seen that the contribution of the PD controller can be found in Figs. 17 and 18. Thus, the second method is effective for suppressing a relatively small effect due to the mapping error of NN-based controller or the tolerable change of mass of the robot as an external disturbance, as shown in Table 4.

The third method was composed of the NN-based feedforward controller, the PD feedback controller and an adaptive fuzzy compensator. This method was able to cover the drastic change of mass of the actual robot. Thus, as can be found from Table 4, this method was able to suppress the position and azimuth errors for all cases considered here by adding an adaptive fuzzy compensator.

6 Conclusions

By applying the concept of a virtual master-slave system, we have described a design method for constructing an NN-based feedforward controller in the framework of neuro-interface for a nonholonomic mobile robot, in which it was assumed that the dynamical and kinematic parameters, except for the offset distance of the center of mass from the mid point of the axle d, are all unknown.

For a practical mapping error, a PD feedback controller was added to the neuro-interface to suppress the output deviations in the sense of two-dof design. However it should be noted that such a PD controller is effective for a fixed or slowly time-varying environment such as a case where there are no sudden changes of mass of the robot. That is, the conventional PD feedback controller was simple to implement, but it is not so powerful against any variational deviations caused by the changes of the robot's mass or against any external disturbances.

To overcome such problems, we further investigated a method for adding a flexible feedback compensator, such as an adaptive fuzzy compensator to the above control system. The effectiveness of the proposed method was shown through some simulations for solving a trajectory tracking control problem of a nonholonomic mobile robot with two-independent driving wheels.

As can be seen from the design of a PD feedback controller or of an adaptive fuzzy compensator, the present research was concentrated on acquiring the adaptation or robustness in the control system, expecting the convergence of

the error $\boldsymbol{e}(t) \stackrel{\triangle}{=} \boldsymbol{q}_r(t) - \boldsymbol{q}(t)$ via simulations. On the contrary, Fierro and Lewis [7], [11], [12] studied a method using a backstepping approach plus a neural network compensation for trajectory tracking problem, path following problem, and point stabilization problem. For the trajectory tracking problem in their methods, the convergence of the error $\boldsymbol{e}(t)$ as well as the convergence of $\boldsymbol{e}_v(t) \stackrel{\triangle}{=} \boldsymbol{v}_r(t) - \boldsymbol{v}(t)$, as $t \to \infty$, were guaranteed by the Lyapunov stability theory. Therefore, it may be very interesting to combining a backstepping approach and a fuzzy compensation, as a future work in a neuro-interface.

References

1. B. Widrow and G. L. Plett, Nonlinear adaptive inverse control, in *Proc. of the 36th Conf. on Decision and Control*, San Diego, California USA, December 1997.
2. W. Widrow and M. L. Lamego, Neurointerfaces, *IEEE Trans. on Control System Technology*, vol. 10, no. 2, pp. 221–228, March 2002.
3. K. Izumi, R. Syam, and K. Watanabe, Neural network based disturbance canceller with feedback error learning for nonholonomic mobile robots, in *Proc. of the 4th Int. Symposium on AI System (ISIS 2003)*, Sept. 25–28, 2003, Jeju, Korea, pp. 443–446.
4. C. Kambhampati, R. J. Craddok, M. Tham, and K. Warwick, Inverse model control using recurrent networks, *Mathematics and Computers in Simulation*, vol. 51, pp. 181–199, 2002.
5. R. Syam, K. Watanabe, and K. Izumi, A Study on Constructing a Neuro-interface Using the Concept of a Virtual Master-Slave System, *Artificial Life and Robotics*, vol. 9, pp. 51–57, 2005
6. R. Syam, K. Watanabe, and K. Izumi, A neurointerface using a concept of virtual master-slave for controlling nonholonomic mobile robots, in *Proc. of Int. Workshop on Fuzzy Systems & Innovational Computing 2004 (FIC2004)*, June 2–3, 2004, Kitakyushu, Japan.
7. R. Fierro and F. L. Lewis, Control of nonholonomic mobile robot using neural networks, *IEEE Trans. on Neural Networks*, vol. 9, no. 4, pp. 589–600, 1998.
8. M. Teshnehlab and K. Watanabe, *Intelligent Control Based on Flexible Neural Networks*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999, pp. 47–49.
9. K. Watanabe, T. Fukuda, and S. G. Tzafestas, An adaptive control for CARMA systems using linear neural networks, *Int. J. Control*, vol. 56, no. 2, pp. 483–497, 1992.
10. Li-Xin Wang, *Adaptive Fuzzy Systems and Control*, Prentice Hall, Englewood Clis, New Jersey, 1994.
11. R. Fierro and F. L. Lewis, Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics, *Journal of Robotic Systems*, vol. 14, no. 3, pp. 149–163, 1997.
12. R. Fierro and F. L. Lewis, Robust practical point stabilization of a nonholonomic mobile robot using neural networks, *Journal of Intelligent and Robotic Systems*, vol. 20, pp. 295–317, 1997.